

SOUTHWEST FISHERIES CENTER

NATIONAL MARINE FISHERIES SERVICE

SOUTHWEST FISHERIES CENTER

P.O. BOX 271

LA JOLLA, CA 92038

LIBRARY

MARCH 1989

APR 27, 1989

National Marine Fisheries Service
2570 Dole Street
Honolulu, Hawaii

A PERSONAL COMPUTER BASED SYSTEM FOR ANALOG-TO-DIGITAL AND SERIAL COMMUNICATION DATA ACQUISITION

by

Robert C. Holland

ADMINISTRATIVE REPORT LJ-89-09

This Administrative Report is issued as an informal document to ensure prompt dissemination of preliminary results, interim reports and special studies. We recommend that it not be abstracted or cited.

**A PERSONAL COMPUTER BASED SYSTEM FOR ANALOG-TO-DIGITAL
AND SERIAL COMMUNICATION DATA ACQUISITION**

by

Robert C. Holland
Southwest Fisheries Center
National Marine Fisheries Service
P.O. Box 271
La Jolla, CA. 92038

March 1989

TABLE OF CONTENTS

INTRODUCTION	1
INSTALLATION	2
EXECUTION	2
OPERATION	3
Standard Mode	3
Waiting Mode	3
Blank Mode	3
INFORMATION	3
Configuration	4
BAUD	4
COMM	4
FORM	4
DIRC	4
ADDR	4
GAIN	4
TICK	4
PCPLUSS Files	4
CONFIGUR.EXE	4
PCPLUSS.BAS	5
PCPLUSS.EXE	5
PCPLUSS.INF	5
PCPLUSS.WP5	5
AIO8	5
EXP-16	6
ODEC Thermosalinograph	6
Output	6
ACKNOWLEDGEMENTS	7
REFERENCES	7
APPENDIX 1 - Specifications	8
Fluorometer	8
EXP-16 Multiplexer	8
AIO8-B Analog-Digital Board	9
APPENDIX 2 - Program PCPLUSS.BAS	10
APPENDIX 3 - Program INSTALL.PAS	18
APPENDIX 4 - Program CONFIGUR.PAS	23

INTRODUCTION

PCPLUSS is a compiled BASIC¹ program designed to sample, average and store multiple streams of data from a variety of sources. It is presently configured to directly access an **AIO8**² analog-to-digital (A/D) board and convert analog data from a **Turner Designs**³ Fluorometer (flow-thru model, series 10) to a 12-bit digital form, and to collect temperature and salinity data from an **ODEC** Thermosalinograph through the communications port. The input data are averaged and stored as: date, time, temperature, salinity, fluorescence voltage, fluorometer scale, and mode of operation. The program can be modified to handle other data inputs, such as **SATNAV** latitude and longitude.

Data are collected at 1 second intervals and are averaged over 2 minutes (default). The data file is opened only when data is to be written, thereby safeguarding data integrity from power loss or equipment malfunction. The **AIO8** is an 8-bit, XT style board that uses an expansion slot in a PC (Personal Computer). The **AIO8** can be attached to an **EXP-16**⁴ multiplexer board through an external cable to allow multiple analog inputs.

The **EXP-16** can be cascaded with 7 other multiplexer boards to provided a maximum of 128 current or voltage analog inputs. Turbo BASIC can operate 2 communication ports (COM1 and COM2). Thus, **PCPLUSS** can control 130 data inputs, although the 1 second interval may not be possible on a computer with an effective clock rate of less than 8 MHz. (megahertz).

PCPLUSS has been executed successfully with 4 inputs (3 analog, 1 serial) sampled at 1 second intervals on a 4.77 MHz. COMPAQ portable. A math coprocessor (Intel 8087/80287) must be used since high-speed math calculations are required in **PCPLUSS**. An AT class computer (80286/80386 microprocessor) with an effective clock rate of 12 MHz. (or better) should be sufficient for any array of data inputs.

PCPLUSS should be installed on a hard disk with an average seek time of 65 ms. (milliseconds) or better. **PCPLUSS** can be installed on a bootable floppy diskette, but floppy drives are extremely slow and are often less reliable than a hard disk (or hard card).

¹Turbo BASIC 1.1. Copyright Borland International, Inc.

²Industrial Computer Source, San Diego, CA. 92123.

³Turner Designs, Mountain View, Ca. 94043.

⁴Industrial Computer Source, San Diego, CA. 92123.

INSTALLATION

To install the PCPLUSS program, turn on the computer and let the computer boot off the hard disk. Place the PCPLUSS INSTALLATION DISK in drive A: and switch to the drive by typing: A:↵. Then type: INSTALL↵. (↵ = <ENTER>)

The INSTALL program will then ask for the BOOT drive:

BOOT Drive: C

The "C" drive can be chosen by pressing <ENTER>, or you can choose a different boot drive by pressing the corresponding key (It is very unlikely that a boot drive will not be "C". Make sure if you choose a different drive, that it is the ACTUAL boot drive).

Then the INSTALL program will ask for the PCPLUSS drive:

PCPLUSS Drive: C

Again, "C" drive can be chosen by pressing <ENTER>, or you can choose a different drive (if the hard disk has only one partition, just press <ENTER>).

The INSTALL program will then ask for the PCPLUSS directory:

PCPLUSS Directory: \PCPLUSS

Press <ENTER> if you want to select the default directory (\PCPLUSS), or enter the new directory and press <ENTER>.

The INSTALL program will then copy the necessary files to the PCPLUSS drive and directory, create the PCPLUSS information file, "PCPLUSS.INF", and add the necessary "SET" and "PATH" commands to the boot drive's AUTOEXEC.BAT file (and save the old file as AUTOEXEC.BAK).

EXECUTION

In the PCPLUSS directory, there should be the files:

- CONFIGUR.EXE - Reconfigures the PCPLUSS.INF file.
- PCPLUSS.BAS - The source program.
- PCPLUSS.INF - The configuration file.
- PCPLUSS.EXE - The executable program.
- PCPLUSS.WP5 - This document in WordPerfect 5.0 format.

To run the program PCPLUSS, just type PCPLUSS↵, i.e.

?> PCPLUSS↵

The program will locate the **PCPLUS.INF** information file and the data files will be written to the pre-configured directory.

OPERATION

When the program is running, it is in the **Standard Mode** (record type = 1) of data acquisition. The date, time, temperature, salinity, fluorometer voltage, fluorometer scale, and record type indicator are constantly shown and updated. Also, the output data file name is shown.

In the **Standard Mode**, there are three function keys available:

- F1** - Suspend Data Acquisition
- F3** - Record Calibration Value
- F10** - Terminate Program

Pressing **F1** places you in the **Waiting Mode**, where only two function keys are now available:

- F1** - Restart Data Acquisition
- F2** - Record Blank Value

Pressing **F1** places you back in the **Standard Mode**, while pressing **F2** places you in the **Blank Mode**, which will run data acquisition for 30 seconds (record type = 2) and place you back in the **Waiting Mode**. This will continue until you press **F1** to go back to the **Standard Mode**. The **Blank Mode** is used to blank the fluorometer at one or more scale settings (Smith et al., 1981).

In the **Standard Mode**, pressing **F3** will clear the function keys and mark the current data record as a calibration record (record type = 3). **F3** is pressed when a water sample is collected from the fluorometer for discrete chlorophyll analysis and used to later calibrate the data collected by **PCPLUS** (Smith et al., 1981). At the start of the next data acquisition interval, **PCPLUS** will return to the **Standard Mode**.

In the **Standard Mode**, pressing **F10** will immediately terminate the program and place you back at the **DOS** (Disk Operating System) prompt.

INFORMATION

The default **PCPLUS.INF** configuration settings are:

```
BAUD:COM1:300,E,7,2
COMM:014,000,001,005,007,005
FORM:\  \  \  \ ##.## ##.## #.### \  \ #
```


DIRC:C:\PCPLUSS
ADDR:768
GAIN:2
TICK:120

Configuration

BAUD specifies the communications port (COM1), the baud rate (300), parity (Even), data bits (7), and the number of stop bits (2).

COMM specifies the number of characters for 1 serial input (014), the ASCII start character (000), the position of the beginning character of temperature after the start character (001), the number of characters representing temperature (005), the position of the beginning character of salinity after the start character (007), and the number of characters representing salinity (005). Each part (6 parts) MUST be present and each part MUST be 3 characters long, each separated by a comma. The current BAUD and COMM strings represent the output of the ODEC thermosalinograph (see ODEC Thermosalinograph for more information on the ODEC output). These strings must be changed for other forms of serial data input.

FORM specifies the TIME and DATE fields (\ \ \ \), the temperature and salinity fields (##.## ##.##), the fluorometer voltage (###.###), the fluorometer scale (\ \), and the blanking indicator (#) (see Output for more information).

DIRC specifies the current directory where the data will be placed when collected (C:\PCPLUSS). This is determined during installation.

ADDR specifies the AIO8 dip switch setting for the Input/Output port (768). The ADDR value in PCPLUSS.INF MUST be the same on the AIO8 board to insure correct input values. This dip switch setting on the AIO8 board should only be changed by a knowledgeable technician with the proper AIO8 manual.

GAIN specifies the gain setting on the EXP-16 multiplexer board. This setting is used on all input signals. The GAIN value in PCPLUSS.INF MUST be the same on the EXP-16 board to insure correct input values. This dip switch setting on the EXP-16 board should only be changed by a knowledgeable technician with the proper EXP-16 manual.

TICK specifies the number of seconds to average the input data (120). This does not change the blanking period (30 seconds).

PCPLUSS Files

CONFIGUR.EXE allows you to change any of the above settings

without changing the program itself. To run the program, just type: **CONFIGUR** (you do not have to be in the **PCPLUSS** directory to run **CONFIGUR**). Then enter the number corresponding to the line you wish to edit. When finished, just press **<ENTER>**. **CONFIGUR.EXE** will locate and change the **PCPLUSS.INF** file in the **PCPLUSS** directory.

PCPLUSS.BAS is the source program in Turbo BASIC. If a major problem occurs where the program needs to be changed, consult a programmer before any changes are made, and the executable program is changed.

PCPLUSS.EXE is the executable program that collects the data. To run, just type: **PCPLUSS** (you do not have to be in the **PCPLUSS** directory to run **PCPLUSS**).

PCPLUSS.INF is the configuration file that contains the changeable input/output and timing parameters of **PCPLUSS**.

PCPLUSS.WP5 is the document you are reading now. It was written using **WordPerfect 5.0**.

AI08

There are 3 channels currently being accessed by the **AI08** on the **EXP-16**: 0) Fluorometer Scale, 1) Fluorescence Voltage, and 2) Voltage of x1/x100 knob. All channels have inputs ranging from 0 to 1 volt. Thus the calculations in the program for the on-screen and in-file data are calculated as:

$$(\text{INPUT [digital]} * \frac{5 \text{ [volts]}}{2048 \text{ [digital]}} - 5) / 2$$

The input from the **AI08** board is in digital form representing a 10 volt range (= 12 bits). This number is then multiplied by 5 (volts) and divided by its digital equivalent 2048 (to put the equation into a volt unit). This value is then subtracted by 5 (to put the whole equation into a -5 to 5 volt range (see table)). The 2 is for the gain setting on the **EXP-16** board.

<u>Binary</u>	<u>Hex</u>	<u>Analog Input Voltage</u>
0000 0000 0000	000	-5.0000 v (-Full scale)
0000 0000 0001	001	-4.9976 v
0100 0000 0000	400	-2.5000 v (-½ scale)
1000 0000 0000	800	±0 v (zero)
1000 0000 0001	801	+0.0024 v
1100 0000 0000	C00	+2.5000 v (+½ scale)
1111 1111 1111	FFF	+4.9976 v (+Full scale)

EXP-16

The EXP-16 has 8 channels (0 - 7) which can be multiplexed through the AIO8 board. Each channel has "signal hi" and "signal lo" inputs for devices with grounded signals. Also, there is an EXP-16 ground associated with each channel for floating signals (all fluorometer inputs have grounded (return) signals). There may need to be a resistor (load) across each "hi" and "lo" channel to scale each input signal to a proper voltage range.

Each channel is then multiplexed through gain circuitry for input signal amplification. The EXP-16 has 8 gain settings: 0.5, 1, 2, 10, 50, 100, 200, 1000. Since the fluorometer inputs are in the range 0 - 1 volt, a gain setting of 2 can be used for each signal (since the AIO8 can only digitize -5 to 5 volts, a gain setting of 10 can not be possible).

ODEC Thermosalinograph

The temperature and salinity are accessed from the ODEC thermosalinograph device through the communications port at baud 300, 7 data bits, even parity, and two stop bits. The form of the input is as follows:

```
<null>TT.TT<sp>SS.SS<cr><lf>  <= 014 characters
```

where <null> is the null character 000 (start character), TT.TT is the input temperature which begins 001 character after the start character and is 005 characters long, <sp> is a space, SS.SS is the input salinity which begins 007 characters after the start character and is 005 characters long, <cr> is a carriage return, and <lf> is a line feed. The ODEC sends the data constantly, but the program will only access the data once a second (and empty the communications buffer for new data). With the above information, the line in the PCPLUSS.INF file is: COMM:014,000,001,005,007,005.

Output

The averaged data are stored in a data file in a format specified by FORM:

```
MMDD HH:MM:SS TT.TT SS.SS F.FFF SSSS B
```

where MMDD is the month and day of the current file, HH:MM:SS is the hours, minutes and seconds, TT.TT is temperature, SS.SS is salinity, F.FFF is fluorescence voltage, SSSS is the scale set on the fluorometer with the following possible scale values:

```
1A  = x1    x1
1B  = x1    x3.16
1C  = x1    x10
```


1D = x1 x31.6
100A = x100 x1
100B = x100 x3.16
100C = x100 x10
100D = x100 x31.6

and B is the record type where:

1 = Normal mode
2 = Blank (blanking mode)
3 = Calibration

The output file name is in the form:

FYYMMDD.DAT

where F stands for File, YY is the year, MM is the month, and DD is the day (the extension .DAT refers to it as a data file).

In a normal day's operation, a file will be about 28Kb in size. Therefore, when backing up the data, about 12 days of data can fit onto one 360Kb-5¼ inch diskette.

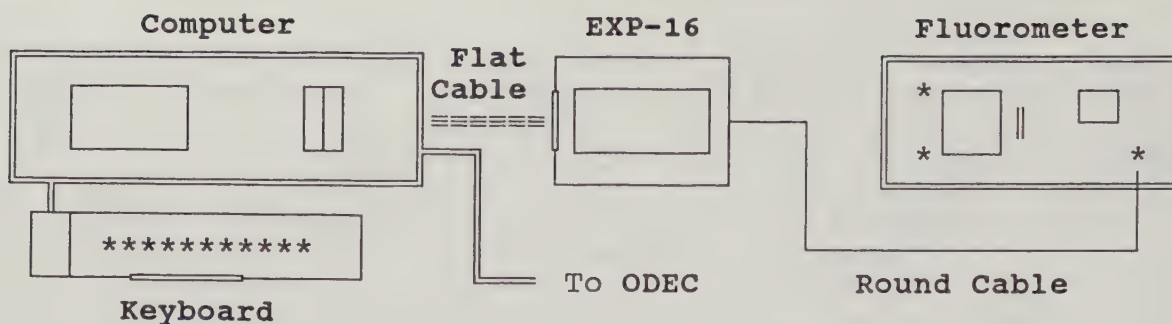
ACKNOWLEDGEMENTS

I would like to thank Dr. Steven Reilly and Dr. Paul Fiedler for their guidance and insight into both the hardware and software aspects of this project. Their continued support engendered this project to be both viable and successful.

REFERENCES

Smith, R.C., K.S. Baker, and P. Dustan. 1981. Fluorometric techniques for the measurement of oceanic chlorophyll in the support of remote sensing. SIO Ref. 81-17, 14 pp.

APPENDIX 1 - Specifications



Fluorometer

Description	Pin	Voltage Range
Analog Output Signal	Pin M (w/ 1k resistor)	0 - 1 Volt
Analog Range Output	Pin N (w/ 1k resistor)	0.0v = x0 0.4v = x3.16 0.7v = x10 1.0v = x31.6
Analog Precision Ground	Pin L	
TTL x1/x100 Knob	Pin S (w/ 1.1k resistor)	0.0v = x100 0.5v = x1
TTL Return Ground	Pin D	

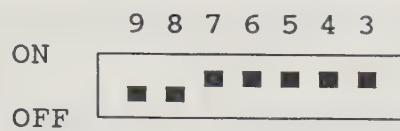
EXP-16 Multiplexer

Input (fluorometer)	Channel (EXP-16 Board)
Pin N	Channel 0 (hi)
Pin M	Channel 1 (hi)
Pin S	Channel 2 (hi)
Pin L	Channel 0 (lo) Channel 1 (lo)
Pin D	Channel 2 (lo)

- 1) 1k resistor across channel 0 (hi) and channel 0 (lo)
- 2) 1k resistor across channel 1 (hi) and channel 1 (lo)
- 3) 1.1k resistor across channel 2 (hi) and channel 2 (lo)
- 4) Toggle switch on gain control set to gain of 2

AI08-B Analog-Digital Board

AI08 board set on base address of 768 (300 Hexadecimal).



APPENDIX 2 - Program PCPLUS.BAS

Program used to collect input data from sensors
(written in Turbo BASIC 1.1 (Borland Intl.))

```
*****
*
*      PCPLUSS.BAS (V. 3.0) - DATA LOGGING PROGRAM FOR AIO8 BOARD
*
*
*      Version 1   06-06-87
*      Version 2   06-24-88
*      Version 3   03-14-89
*****
```

```
----- Description of Program -----
This program sets up the AIO8 to log data from the first 3 analog
channels (no digital inputs): 0) Fluorometer scale, 1) Fluorescence,
2) Voltage of x1 & X100 knob. The data is transferred to a random
access data file on disk at 2 minute intervals.
```

```
DIM DI%(3)           'Dimension channel data array
```

Set up EVENT traps

```
ON KEY(1) GOSUB Waiting      'Function key 1 waits for blanking
ON KEY(2) GOSUB BlankOn     'Function key 2 turns blanking ON
ON KEY(3) GOSUB Calibrate   'Function key 3 sets calibration ON
ON KEY(10) GOSUB GoodNight  'Ends program with Function key 10
CLS
```

Get Program Information from information file

```

BAUD$ = "COM1:300,E,7,2"      'Initial Port, Baud, Parity, Data, Stop
COMM$ = "014,000,001,005,007,005" 'Initial temp/sal input form
FORM$ = "\ \ \ \ \ ##.## ##.## #.### \ \ #" 'Initial Output image
DIRC$ = ""                    'Initial output file directory string
ADDR% = 768                    'Initial Base address for AIO8 board
GAIN% = 2                      'Initial EXP-16 gain setting
TICK% = 120                    'Initial seconds count between writes

```

```
PCPLUS$ = ENVIRON$("PCPLUS") + "\PCPLUS.INF"
OPEN PCPLUS$ FOR APPEND AS #1
CLOSE #1
OPEN PCPLUS$ FOR INPUT AS #1      'Open PCPLUS information file
WHILE NOT EOF(1)
  LINE INPUT #1,INSTRING$        'Get input string from input file
```



```

T$ = MID$(INSTRING$,6,LEN(INSTRING$) - 5)
SELECT CASE LEFT$(INSTRING$,5)
  CASE "BAUD:"
    BAUD$ = T$ 'If baud input, set baud string
  CASE "COMM:"
    COMM$ = T$
  CASE "FORM:"
    FORM$ = T$ 'If form input, set form string
  CASE "DIRC:"
    DIRC$ = T$ + "\" 'If directory input, set DIRC string
  CASE "ADDR:"
    ADDR% = VAL(T$) 'If PCPLUSS address, set address port
  CASE "GAIN:"
    GAIN% = VAL(T$) 'If gain input, then set gain value
  CASE "TICK:"
    TICK% = VAL(T$) 'If seconds tick, set TICK variable
END SELECT
WEND
CLOSE #1

```

```

NUMCHARS% = VAL(MID$(COMM$,1,3)) 'Number of characters from TempSal
STARTCHAR% = VAL(MID$(COMM$,5,3)) 'Character that signifies start
TEMPCHAR% = VAL(MID$(COMM$,9,3)) 'Position of temp after STARTCHAR%
TEMPLLENGTH% = VAL(MID$(COMM$,13,3)) 'Number of characters for Temperature
SALCHAR% = VAL(MID$(COMM$,17,3)) 'Position of sal after STARTCHAR%
SALLENGTH% = VAL(MID$(COMM$,21,3)) 'Number of characters for Salinity
NUMCHARS% = NUMCHARS%*2 'Double NUMCHARS% so to include 1 line
OPEN BAUD$ + ",DS,RS,CS,CD" FOR INPUT AS #3 'Open COM port for TS input
$COM1 2048 'Set up 2k buffer for COM 1 port

```

Turn keys ON to allow traps to happen

```

KEY(1) ON 'Turn Waiting key F1 ON
KEY(3) ON 'Turn Calibrate key F3 ON
KEY(10) ON 'Turn Exit key F10 ON

```

Determine Screen characteristics (Monochrome or Color)

```

CLS
REG 1,&H0F00 'Set register AH with 0F
CALL INTERRUPT &H10 'Call BIOS interrupt for screen ID
IF (REG(1) AND &H00FF) <> 7 THEN 'If AL=7 then Monochrome, else color
  WIDTH 40 'Set up screen width to 40 characters
END IF ' if in color mode (AL <> 7)

```

Get name of data file

```

ODAT$ = DATE$                                ' if file already exists
FILE$ = "F" + MID$(ODAT$,9,2) + LEFT$(ODAT$,2) + MID$(ODAT$,4,2) + ".DAT"
OPEN DIRC$ + FILE$ FOR APPEND AS #1'Create the data file then close
CLOSE #1
ON ERROR GOTO Errorcom                        ' to occur with a communication error

```

Initialize Screen with colors and text

```

CLS
LOCATE 24,1
COLOR 5,0
PRINT "PCPLUSS (3.0) - Data Acquisition Program"
COLOR 7,0
LOCATE 3,1
PRINT "  Date          Time (GMT)"
PRINT "  -----          -----"
PRINT
PRINT
PRINT "  TEMP  SAL  FLUOR  SCALE  B"
PRINT "-----  -----  -----  -----  -"
LOCATE 12,1
PRINT "File: ";DIRC$ + FILE$

```

Start:

```

LOCATE 17,1
COLOR 0,7
PRINT " - <F1>  Suspend data acquisition  - "
PRINT "                                           "
PRINT " - <F3>  Record calibration value   - "
PRINT " - <F10> Terminate program           - "
COLOR 7,0

```

Initialize Working Variables

COL% = 0	'Variable for count of samples taken
TSCOL% = 0	
BLANK% = 1	'Blanking occurs when BLANK% = 0
FLUOR = 0	'Set fluor, temp, and sal to 0 for
TEMP = 0	' sampling
SAL = 0	
T = 0	'Set temp and sal values to 0
S = 0	
SECONDS% = TICK%	'Number of seconds to sample data
BLANK\$ = " SAMPLING "	'Set flashing indicator to "SAMPLING"
BLKPRESSED% = 1	'Indicator for blanking key pressed
OLDTIME\$ = TIME\$	'Save current time for checking later

Fetch A/D data from channels 0 thru 2 and return in DI%(*)

```
inLoop:
EVENT OFF                                'Temporarily turn off all event traps
COLOR 0,7
LOCATE 15,16
PRINT BLANK$                             'Print "SAMPLING" with color
COLOR 7,0

FOR I% = 0 TO 2
  ADDRESS% = I%*16                       'Set multiplexer channel to 0 and
  OUT (ADDR% + 2),ADDRESS%               ' use analog channels 0 - 2
  OUT (ADDR% + 1),0                      'Start 12-bit conversion
  XL% = INP(ADDR%)                       'Get low byte of data
  XH% = INP(ADDR% + 1)                   'Get high byte of data
  DI%(I%) = XH%*16 + XL%\16             'Combine high and low bytes
NEXT I%
INCR COL%                               'Increment collector for sampling #
```

Get time and date

```
TI$ = TIME$
DAT$ = DATE$
DAT$ = MID$(DAT$,1,2) + "/" + MID$(DAT$,4,2) 'Set in MM/DD
```

Display current data

```
C0 = (DI%(0)*5!/2048! - 5)/GAIN%         'Convert digital data to voltage
C1 = (DI%(1)*5!/2048! - 5)/GAIN%         ' representation of input values
C2 = (DI%(2)*5!/2048! - 5)/GAIN%
IF C1 < 0 OR C1 > 11 THEN C1 = 0         'In case fluorometer is not working
FLUOR = FLUOR + C1                       'Add latest value of fluorescence and
C1 = FLUOR/COL%                          ' average over the number collected
```

Determine x1/x100 knob setting and fluorometer scale

```
IF C2 > 0.25 THEN SCALE$ = "1" ELSE SCALE$ = "100"
SELECT CASE C0
  CASE > 0.85
    SCALE$ = SCALE$ + "D"
  CASE > 0.55
    SCALE$ = SCALE$ + "C"
  CASE > 0.2
    SCALE$ = SCALE$ + "B"
```

```

CASE ELSE
    SCALE$ = SCALE$ + "A"
END SELECT
IF COL% = 1 THEN OLDSCALE$ = SCALE$

```

Get temperature and salinity from the tempsal (serial port)

```

IF LOC(3) > NUMCHARS% THEN      'If NUMCHARS% are present then
    TEMPSAL$ = INPUT$(LOC(3),#3) ' get those characters
    POSITION% = INSTR(1,TEMPSAL$,CHR$(STARTCHAR%)) 'Find the start
    TEMP = TEMP + VAL(MID$(TEMPSAL$,POSITION% + TEMPCHAR%,TEMPLLENGTH%))
    SAL = SAL + VAL(MID$(TEMPSAL$,POSITION% + SALCHAR%,SALLENGTH%))
    INCR TSCOL%                  'Increment the temp sal counter
    T = TEMP/TSCOL%              'Get the on-going temperature and
    S = SAL/TSCOL%              ' salinity for the screen
END IF

```

Output the data to screen

```

LOCATE 5,1
PRINT USING " \ \ \ \";DAT$;TI$
LOCATE 9,1
PRINT USING "##.## ##.## #.### \ \ #";T;S;C1;SCALE$;BLANK%

```

Write data to disk

```

$EVENT ON      'Turn on event trap for keyboard keys
IF (COL% < SECONDS%) AND (BLKPRESSED%) AND (SCALE$ = OLDSCALE$) THEN EndScan
IF SCALE$ <> OLDSCALE$ THEN      'Has the scale changed since last
    C1 = OLDC1                  ' sample? If so, then get the last
    SCALE$ = OLDSCALE$         ' value of fluorescence
END IF
COL% = 0        'Reset the Fluoresence and tempsal
TSCOL% = 0      ' counters to 0 for more data
OPEN DIRC$ + FILE$ FOR APPEND AS #1 'Temporarily open file for output
PRINT #1,USING FORM$;DAT$;TI$;T;S;C1;SCALE$;BLANK% 'Write data to file
CLOSE #1       ' then close the file immediately
IF BLANK% = 3 THEN      'If in calibration mode, then reset
    BLANK% = 1          ' back to sampling mode, and turn
    KEY(1) ON           ' trap keys back on
    KEY(3) ON
    KEY(10) ON
    GOTO Start
END IF
IF BLKPRESSED% = 0 THEN WaitBlank 'If the Blank key has been pressed or
IF SECONDS% = 30 THEN WaitBlank  ' 30 seconds elapsed, go to WaitBlank

```



```

rReset:
COL% = 0
TSCOL% = 0
FLUOR = 0
TEMP = 0
SAL = 0
'Reset input variables and collection
' counters

dScan:
OLDSCALE$ = SCALE$
OLDC1 = C1
LOCATE 15,16
PRINT SPC(20)
'Make old scale equal new scale and
' old fluorescence to new fluorescence
'Remove sampling marker

```

Delay for sampling interval

```

DO
LOOP UNTIL OLDTIME$ <> TIME$

OLDTIME$ = TIME$
C$ = INKEY$
'Remove any unwanted keys pressed

```

Loop back for next data

```

GOTO MainLoop

```

Event Key Traps Modules Begin here (Trap routines are further below)

```

itBlank:
BLKPRESSED% = 1
BLKON% = 0
KEY(1) ON
KEY(2) ON
KEY(3) OFF
KEY(10) OFF
LOCATE 17,1
COLOR 0,7
PRINT " - <F1> Restart data acquisition - "
PRINT " - <F2> Record blank value - "
PRINT " "
PRINT " "

LOCATE 15,16
PRINT " WAITING "
COLOR 7,0

```

```

Waitloop:
  OLDTIME$ = TIME$
  DO
    INTTIME$ = TIME$
  LOOP UNTIL INTTIME$ <> OLDTIME$
  IF LOC(3) > 0 THEN TEMPSAL$ = INPUT$(LOC(3),#3) 'Clear buffer
  IF BLKON% THEN Blanking 'If <F2> pressed then start Blanking
  IF BLKPRESSED% THEN Waitloop 'If <F1> wasn't pressed then wait
  BLKPRESSED% = 1 'Reset <F1> pressed variable
  KEY(1) ON 'Restore previous key configurations
  KEY(2) OFF
  KEY(3) ON
  KEY(10) ON
  GOTO Start

```

```

Blanking:
  KEY(1) OFF 'Disable <F1> and <F2> while blanking
  KEY(2) OFF
  LOCATE 17,1
  COLOR 0,7
  PRINT "
  PRINT "
  COLOR 7,0
  SECONDS% = 30 'Blank for 30 seconds
  BLANK% = 2 'Show that program is in blanking mode
  BLANK$ = " BLANKING " ' and flash a "BLANKING" message
  GOTO VarReset

```

The following are keyboard, timer and error traps.

Waiting is for <F1> key trap
 BlankOn is for <F2> key trap
 Calibrate is for <F3> key trap
 ErrorCom is for communications error trap
 GoodNight is for <F10> key trap

```

Waiting:
  BLKPRESSED% = 0 'If <F1> pressed, set the flag to 0
  RETURN

```

```

BlankOn:
  BLKON% = 1 'If <F2> pressed, set the flag to 0
  RETURN

```

```

Calibrate:
  BLANK% = 3 'Set mode to calibration = 3
  KEY(1) OFF 'Turn off all keyboard traps
  KEY(3) OFF
  KEY(10) OFF
  LOCATE 17,1

```



```
COLOR 0,7
FOR II% = 1 TO 4
  PRINT "
NEXT II%
COLOR 7,0
RETURN
```

```
Errorcom:                                'Don't ask why an error occurred, just
RESUME                                    ' go back and try again
```

```
GoodNight:                                'With <F10> pressed, terminate the
LOCATE 24,1                                ' program
COLOR 0,7
PRINT" TERMINATED ";
COLOR 7,0
LOCATE 22,1
CLOSE #1
END
```

APPENDIX 3 - Program INSTALL.PAS

Program used to install the necessary files and system parameters.
(written in Turbo Pascal 5.0 (Borland Intl.))

```
program Install;
uses Crt, Dos;

{ Files contains the names of the files to be copied }
const NumFiles = 4;
      Files : array[1..NumFiles] of string[13] = (' PCPLUSS.EXE',' PCPLUSS.BAS
                                                ' PCPLUSS.WP5'
                                                CONFIGUR.EXE');
var Dirinfo : searchrec;
    S,S1    : string;
    Auto    : array[1..100] of string[150]; { Array for AUTOEXEC.BAT file }
    Drv,Boot : string[2]; { Strings for boot drive and PCPLUSS drive }
    Dir      : string[50]; { String for PCPLUSS directory }
    F        : text;
    Ch       : char;
    A,B,C    : integer;
    Sets     : boolean; { Boolean variable for AUTOEXEC.BAT SET variable }
    Valid    : boolean;

{ Reboots the computer so the system parameters will take effect }
Procedure ReBoot;
begin
  inline(
    $B8/$40/
    $00/$8E/
    $D8/$B8/
    $34/$12/
    $A3/$72/
    $00/$EA/
    $5B/$E0/
    $00/$F0);
end;

{ Copies the necessary files for PCPLUSS }
Procedure CopyFile(A:integer);
var FromF,ToF : file;
    NumRead,NumWritten : word;
    Buf             : array[1..4096] of char;
    S               : string[13];
begin
  gotoxy(12,A + 3); { Draw arrow pointing to file to be copied }
  write('->');
  S := copy(Files[A],2,length(Files[A]) - 1);
  assign(FromF,S1 + '\' + S);
  reset(FromF,1);
  assign(ToF,Drv + Dir + '\' + S);
```



```

rewrite(ToF,1);
repeat
  blockread(FromF,Buf,Sizeof(Buf),NumRead); { Get 1 block of data }
  blockwrite(ToF,Buf,NumRead,NumWritten); { Write 1 block of data }
until (NumRead = 0) or (NumWritten <> NumRead);
close(FromF);
close(ToF);
for NumRead := 30 to 53 do { Do a scroll of the file name across the screen
begin
  gotoxy(NumRead - 1,A + 3);
  write(' ');
  gotoxy(NumRead,A + 3);
  write(Files[A]);
  delay(25);
end;
gotoxy(12,A + 3);
write(' ');
end;

Main program )
begin
  clrscr;
  Sets := false;
  findfirst('INSTALL.EXE',anyfile,dirinfo); { See if on the install disk }
  if Doserror = 0 then
  begin
    S1 := fexpand('INSTALL.EXE'); { Get full file name and directory }
    delete(S1,pos('INSTALL.EXE',S1) - 1,12); { Delete the INSTALL.EXE part }
    repeat
      Boot := 'C: '; { Initialize Boot to the C drive }
      gotoxy(33,9);
      write('BOOT drive: ',Boot); { Get user input }
      gotoxy(45,9);
      Ch:=readkey;
      if upcase(Ch) in ['C','D','E','F'] then
      begin
        write(upcase(Ch));
        Boot[1] := Ch;
      end;
    until upcase(Ch) in ['C','D','E','F',#13];
    for B := 1 to length(Boot) do Boot[B] := upcase(Boot[B]); { Make uppercase

  repeat
    Drv := 'C: '; { Initialize Drv to the C drive }
    gotoxy(30,10);
    write('PCPLUSS drive: ',Drv); { Get user input }
    gotoxy(45,10);
    Ch:=readkey;
    if upcase(Ch) in ['C','D','E','F'] then
    begin
      write(upcase(Ch));
      Drv[1] := Ch;

```

```

end;
until upcase(Ch) in ['C','D','E','F',#13];
for B := 1 to length(Drv) do Drv[B] := upcase(Drv[B]); { Make uppercase }

Valid := false; { Boolean variable for directory validation }
repeat
  Dir := '\PCPLUSS'; { Initialize PCPLUSS directory }
  gotoxy(26,11);
  write('PCPLUSS directory: ',Dir,' ');
  gotoxy(45,11);
  Ch := readkey;
  if Ch <> #13 then { If not a carriage return, then there's more input }
  begin
    write(' ');
    gotoxy(45,11);
    write(Ch);
    readln(S);
    Dir := Ch + S;
    if Dir[1] <> '\' then insert('\',Dir,1);
    if Dir[length(Dir)] = '\' then delete(Dir,length(Dir),1);
    C := 0;
    for B := 1 to length(Dir) do
      begin
        Dir[B] := upcase(Dir[B]); { Make uppercase }
        if Dir[B] = '\' then inc(C); { Count the number of "\" }
      end;
    if C = 1 then Valid := true; { If only 1 "\" then valid }
    end else Valid := true; { else repeat process }
  until Valid;

  findfirst(Drv + Dir,Directory,Dirinfo); { See if directory exists }
  if Doserror <> 0 then mkdir(Drv + Dir); { If not, then create }

```

```

{ Create the visual part with file names and pointers }
clrscr;
gotoxy(34,1);
write('Copying Files');
gotoxy(((20 - length(S1)) div 2) + 10,2);
write(S1);
gotoxy(40,2);
write('To');
gotoxy(((20 - length(Drv+Dir)) div 2) + 50,2);
write(Drv+Dir);
gotoxy(10,3);
write('_____');
gotoxy(50,3);
write('_____');
for A:=1 to NumFiles do
begin
  gotoxy(14,A + 3);
  writeln(Files[A]); { Display file names }
end;

```



```

for A:=1 to NumFiles do CopyFile(A); { Go do the file copying }

gotoxy(1,10);
writeln('Current PCPLUS input file information:');
writeln;
writeln('    BAUD:COM1:300,E,7,2');
writeln('    COMM:014,000,001,005,007,005');
writeln('    FORM:\    \    \    \ ##.## ##.## #.### \    \ #');
writeln('    DIRC:',Drv + Dir);
writeln('    ADDR:768');
writeln('    GAIN:2');
writeln('    TICK:120');
writeln;
writeln('Use CONFIGUR.EXE to change PCPLUS information');
assign(F,Drv + Dir + '\PCPLUS.INF');
rewrite(F);
writeln(F,'BAUD:COM1:300,E,7,2');
writeln(F,'COMM:014,000,001,005,007,005');
writeln(F,'FORM:\    \    \    \ ##.## ##.## #.### \    \ #');
writeln(F,'DIRC:',Drv + Dir);
writeln(F,'ADDR:768');
writeln(F,'GAIN:2');
writeln(F,'TICK:120');
close(F);

gotoxy(1,20);
writeln('Editing AUTOEXEC.BAT file');
findfirst(Boot + '\AUTOEXEC.BAT',Anyfile,Dirinfo); { Locating AUTOEXEC.BAT
if Doserror = 0 then
begin
    assign(F,Boot + '\AUTOEXEC.BAT');
    reset(F);
    A := 0;
    while not eof(F) do { Read in old AUTOEXEC.BAT }
    begin
        inc(A);
        readln(F,Auto[A]);
        if pos('SET PCPLUS',Auto[A]) > 0 then Sets := true;
    end;
    close(F);
    rename(F,Boot + '\AUTOEXEC.BAK'); { Rename to AUTOEXEC.BAK }
    assign(F,Boot + '\AUTOEXEC.BAT'); { Create new AUTOEXEC.BAT }
    rewrite(F);
    for B := 1 to A do
    begin
        if (pos('PATH',Auto[B]) > 0) or (pos('path',Auto[B]) > 0) then
            if pos('PCPLUS',Auto[B]) = 0 then Auto[B] := Auto[B] + ';' + Drv + Dir
            if (not Sets) and (B = 1) then writeln(F,'SET PCPLUS=',Drv + Dir);
        writeln(F,Auto[B]);
    end;
    close(F);
end
end

```

```
else
begin
  assign(F,Boot + '\AUTOEXEC.BAT'); { Create a new AUTOEXEC.BAT }
  rewrite(F);
  writeln(F,'PATH ',Drv + Dir); { Make the PATH to PCPLUS }
  writeln(F,'SET PCPLUS=',Drv + Dir); { Make the SET for PCPLUS }
  close(F);
end;

gotoxy(20,22);
writeln('PCPLUS Installed Successfully!');
gotoxy(20,23);
textcolor(White + Blink);
writeln('Remove Diskette from drive and press Return!');
readln;
reboot;
end else writeln('INSTALL must be executed from the diskette!');
end.
```


APPENDIX 4 - Program CONFIGUR.PAS

Program used to reconfigure the PCPLUS information file.
(written in Turbo Pascal 5.0 (Borland Intl.))

```
Program Configure;
uses Crt,Dos;
var Dirinfo:searchrec;
    Ps      :pathstr;
    IO      :text;
    S       :array[1..20] of string[80];
    Streng  :string[80];
    A,N,C   :integer;

begin
  clrscr;
  Ps := getenv('PCPLUS'); { Locate the system variable for PCPLUS }
  if Ps <> '' then { If found then continue }
  begin
    Ps := Ps + '\PCPLUS.INF';
    findfirst(Ps,Anyfile,Dirinfo); { Locate the PCPLUS.INF file }
    if Doserror = 0 then { If found then continue }
    begin
      assign(IO,Pt);
      reset(IO);
      N := 0;
      while not eof(IO) do { Open file and read in contents }
      begin
        inc(N);
        readln(IO,S[N]);
      end;
      close(IO);
      repeat
        clrscr;
        gotoxy(1,1);
        for A := 1 to N do writeln(A:2,' : ',S[A]); { Write out the contents }
        repeat
          gotoxy(1,22);
          write('Edit #: ');
          gotoxy(9,22);
          A := 0;
          readln(Streng); { Get user input for the line to edit }
          if Streng <> '' then val(Streng,A,C);
        until ((A in [1..20]) and (A <= N)) or (Streng = '');
        if A in [1..20] then { If there was input, then get the new string }
        begin
          gotoxy(3,23);
          write(copy(S[A],1,5)); { Write out the string header, i.e. ADDR: }
          readln(Streng); { Get the new string }
          for C := 1 to length(Streng) do Streng[C] := upcase(Streng[C]);
          if Streng = '' then Streng := S[A] else S[A] := copy(S[A],1,5) + Streng
        end;
      end;
```

```
until Streng = '';
rewrite(IO); { With all of the changes made, output the results }
for A:=1 to N do writeln(IO,S[A]);
close(IO);
end else writeln('Could not find: ',Ps);
end else writeln('PCPLUSS has not been installed!');
end.
```